(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 January 2002 (03.01.2002)

PCT

(10) International Publication Number
WO 02/01365 A2

(51) International Patent Classification[7]: G06F 12/00

(21) International Application Number: PCT/US01/17851

(22) International Filing Date: 1 June 2001 (01.06.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/602,010 23 June 2000 (23.06.2000) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, Santa Clara, CA 95052 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **COULSON, Richard** [US/US]; 17454 NW Gilbert Lane, Portland, OR 97229 (US).

(74) Agent: **CORSELLO, Kenneth, R.**; Kenyon & Kenyon, 333 West San Carlos Street, Suite 600, San Jose, CA 95110 (US).

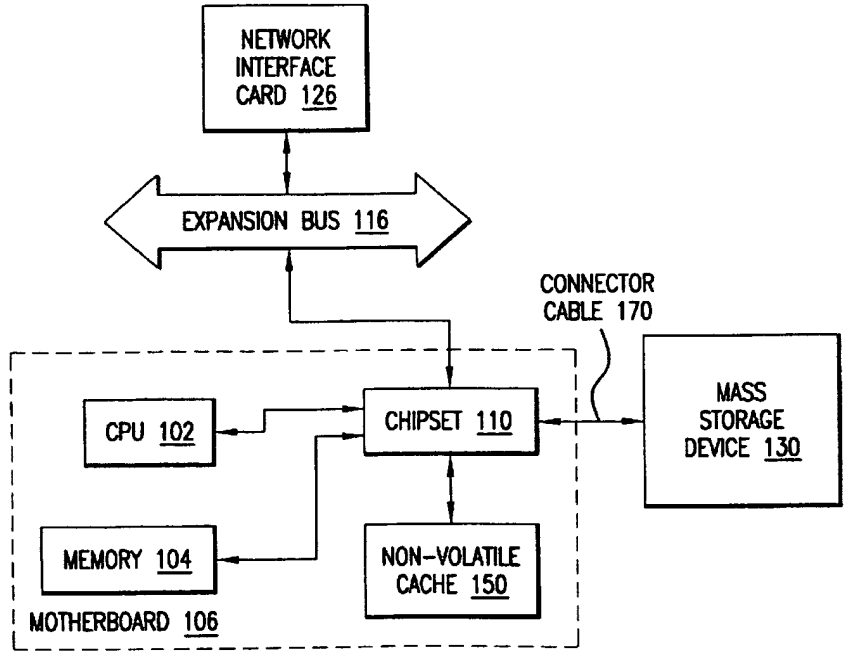
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: NON-VOLATILE CACHE



(57) Abstract: A non-volatile cache including a non-volatile memory to cache data that is stored on a mass storage device.

WO 02/01365 A2

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

NON-VOLATILE CACHE

FIELD OF THE INVENTION

Embodiments of the present invention relate to a non-volatile cache. In particular, the present invention relates to a non-volatile memory to cache data for a disk storage device.

BACKGROUND

Computer systems often store data both in a volatile memory and a non-volatile mass storage device. Such data often includes computer instructions, for operating system and application programs, and data that is accessed by these instructions. A disk drive is an example of a mass storage device that can be part of a modern computer system. Non-volatile mass storage devices, such as disk drives, typically do not lose the data stored on them when the power to the computer system is removed or turned off. Thus, a non-volatile mass storage device such as a disk drive may store data that the computer system is to retain on a relatively permanent basis.

In many computer systems, a set of data currently being used by the computer system's microprocessor may be copied from a disk drive into a volatile random access memory (RAM) because a RAM has a significantly faster access time than a disk drive. However, units of RAM memory (e.g., 1 megabyte) are typically more expensive than that amount of memory in a disk drive, and thus in the typical computer system the storage capacity of the RAM is significantly smaller than the storage capacity of the disk drive. When required data is not present in the RAM memory, the microprocessor can read a block of information including the required data from the disk drive into the RAM. If there is not enough room in the RAM memory, the microprocessor can free space in the RAM by writing a block of data from the RAM back to the disk drive.

Because the access time for a disk drive is generally slower than the access time for RAM, the disk drive is often a performance bottleneck. In addition, accessing data from the disk drive typically uses more power than accessing data from RAM because the disk drive must be mechanically moved. Known disk drives include a volatile cache (e.g.,

a DRAM cache, an SRAM cache), but such volatile caches are typically part of the microprocessor's main memory address space and thereby byte addressed.

DESCRIPTION OF THE DRAWINGS

FIG. 1 is a partial block diagram of a computer system that has a non-volatile cache according to an embodiment of the present invention.

FIG. 2 is a partial block diagram of a disk drive according to an embodiment of the present invention.

FIG. 3 is a partial block diagram of a chipset and non-volatile disk cache according to an embodiment of the present invention.

FIG. 4 is a more detailed partial block diagram of a non-volatile disk cache according to an embodiment of the present invention.

FIG. 5 is a partial block diagram of an entry in a non-volatile disk cache according to an embodiment of the present invention.

FIG. 6 is a flow chart of a method of obtaining data associated with a sector in a disk drive according to an embodiment of the present invention.

FIG. 7 is a partial block diagram of a computer system that has a non-volatile disk cache expansion card according to an embodiment of the present invention.

FIG. 8 is a partial block diagram of a disk drive that has a non-volatile disk cache according to an embodiment of the present invention.

FIG. 9 shows a method of storing data in a non-volatile memory according to an embodiment of the present invention.

DETAILED DESCRIPTION

The methods and apparatus described herein relate to a cache for a mass storage device that can be implemented in non-volatile memory. In particular, embodiments of the present invention relate to a non-volatile cache that can improve the efficiency of memory storage and retrieval by caching data in a non-volatile memory.

FIG. 1 shows a partial block diagram of a computer system ("computer") that has a non-volatile cache according to an embodiment of the present invention. In particular, FIG. 1 shows a computer system 100 that includes a central processing unit (CPU) 102, or processor, and a memory unit 104 mounted to a motherboard 106. The processor 102 may be, for example, a Pentium III processor made by Intel Corporation of Santa Clara, California, an application specific integrated circuit (ASIC), a microcontroller, etc. An example of a memory 104 that may be used in computer system 100 is a 128 megabyte dynamic random access memory device (DRAM). Memory 104 may also be, for example, a ROM. Alternatively, the processor 102 and the memory 104 may reside on separate printed circuit cards that are coupled to the motherboard 106. The term "coupled" encompasses a direct connection, an indirect connection, an indirect communication, etc.

A chipset 110 is coupled to, and manages interaction between, the processor 102 and other system components, such as the memory 104, mass storage device 130 and peripheral components attached to expansion bus 116. As used in this application, the term chipset refers to a group of one or more integrated circuit chips that acts as a hub (or core) for data transfer between the processor and components of the computer system. As shown in FIG. 1, the chipset may be coupled to the computer system's motherboard. Chipset 110 may be integrated with (e.g., soldered on to) motherboard 106. Chipset 110 may be similar to, for example, the 820 and 810E chipsets made by Intel, Corporation of Santa Clara, California. Chipset 110 may be a single integrated circuit or may comprise two or more integrated circuit chips. As shown in FIG. 3, chipset 110 may include a memory control hub (MCH) 311, which performs what is known as "northbridge functionality," and an input/output controller hub (ICH) 312, which performs what is known as "southbridge functionality."

Mass storage device 130 may be a disk drive and may be coupled to chipset 110 through a connector cable 170. Mass storage device 130 may also be a 3.5-inch diskette, a 5.25-inch floppy diskette, a ZIP disk (e.g., manufactured by Iomega Corporation of Roy, Utah), a Jaz disk (e.g., manufactured by Iomega Corporation), an LS-120 Superdisk (e.g., manufactured by Imation Corporation of Oakdale, Minnesota), a rewritable Digital Versatile Disc (DVD-RAM), a Read/Write Compact Disk (CD-RW), optical storage device, magneto-optical storage device, magnetic storage device, holographic, etc. For convenience, this application discloses embodiments in which the mass storage device is a disk drive. An embodiment of a disk drive that may be used with the present invention is shown in FIG. 2.

Expansion bus 116 may be a peripheral component interface (PCI) bus, which is a type of data bus often found in the computer system 100 and which complies with a PCI Local Bus Specification. One or more PCI compliant peripheral devices, such as a network interface card (NIC) 126, may be connected to the expansion bus 116. The network interface card 126 may tie the computer system 100 to a local or wide area computer network.

An embodiment of the present invention is a non-volatile cache that comprises a non-volatile memory to cache data that is stored on a mass storage device. In an embodiment, shown for example in FIG. 1, a non-volatile cache 150 is coupled to chipset 110. In an embodiment, the chipset and non-volatile memory cache data for mass storage device 130. In a further embodiment, the mass storage device is a disk drive and non-volatile cache 150 caches data for the disk drive, in which case the cache may be referred to as a disk cache.

FIG. 2 shows a partial block diagram of disk drive 230 according to an embodiment of the present invention. Disk drive 230 may be the same as mass storage device 130 of FIG. 1, and is assumed to be mass storage device 130 for the purpose of the discussion below. Disk drive 230 may be an external disk drive or an internal disk drive. Disk drive 230 may store, for example, over 100 gigabytes of data. In an embodiment,

disk drive 230 is a hard drive. In a further embodiment, disk drive 230 is a Cheetah 18XL disk drive made by Seagate Technology Inc. of Scotts Valley, California. Disk drive 130 may comprise a disk controller 210 and a mass storage 220. Disk controller 210 may be an application specific integrated circuit and may contain a microprocessor unit (MPU) 211 coupled to a buffer and program memory 215 and ECC logic 218. Buffer and program memory 218 may store a disk drive operation program 219 that is executed by MPU 211 and may also be used as a buffer to store data that is being written to mass storage 220 and read from mass storage 220. MPU 211 may process requests from CPU 102 to write data to or read data from mass storage 220. For example, MPU 211 may determine when a request has been made, and may translate from a logical sector address to a physical sector address. Buffer and program memory 215 may be coupled to disk spindle interface logic 213 which may interface with the host system (e.g., computer system 100). Disk controller 210 may also contain disk formating logic to format data blocks (e.g., insert a preamble and special characters) that are written to the mass storage. As used in this application, "logic" may include hardware logic, such as circuits that are wired to perform operations, or program logic, such as firmware that performs operations.

Mass storage 220 may store, among other things, the computer's operating system (OS) code 231, which, upon boot-up, may be loaded into the computer's memory 104 for execution by the processor 102. Mass storage 220 may also store a device driver 235 for the disk drive, which may also be loaded into the computer's memory 104 for execution by the processor 102 to perform data translation and communication with disk drive 230. The mass storage 220 also may store, among other things, application programs and data that is accessed by these programs. The mass storage 220 may contain platters that may be divided into tracks, which in turn may be divided into sectors. For example, a formatted disk might have 1000's of tracks per platter. In an embodiment, when disk controller 210 receives requests from outside the disk drive to read or write data, the requests may be made in terms of logical disk sectors, and the disk controller may

translate these into physical disk sectors using a mapping algorithm. Some physical sectors may be left out for spares, and the mapping may be hardware assisted.

In operation, when the OS 231 (being executed by processor 102) receives a request for data from a program or other part of the OS, a file system may be used to check a file allocation table (FAT) to determine whether the data is currently in the memory 104, and if not, where on the mass storage 220 (e.g., disk sector address) the data can be found. If the OS 231 determines that the data it needs is not in the memory 104, but is stored on disk drive 230, OS 231 may send a request for the data through chipset 110 to disk drive 230. More precisely, OS 231 may request that disk drive 230 send a copy of the disk sector containing the data that OS 231 requests. Disk controller 210 may then receive the request for the disk sector, retrieve the disk sector from mass storage 220, and send the data for that disk sector through chipset 110 to memory 104. At another time, OS 231 may cause a disk sector of data to be written from memory 104 to disk drive 230, in which case disk controller 210 will receive the data and cause it to be stored on a disk sector of mass storage 220.

In an embodiment, the sector is the smallest addressable physical storage unit used by the disk drive 230, although management information such as addressing information and validity data may be accessed in smaller units. In a further embodiment, each disk sector is the size of 512 bytes of data. In an embodiment, a block is a group of one or more disk sectors that may contain identification codes, error detection codes and/or error correcting codes (ECC) for the block. In an embodiment, the OS requests data from the disk drive in terms of a disk sector or block of sectors (i.e., it would not request a byte or word), in which case the disk drive is said to be block-oriented (i.e., block-addressable). In an embodiment of a block-oriented disk drive, management information may be accessed in smaller units. ECC is an advanced error detection and correction protocol that may detect single-bit and multi-bit errors and may correct some errors on the fly. In an embodiment, the disk sectors in a block are physically located next to each other on one of the disk platters. In a further embodiment, the blocks are manipulated as a unit by disk

driver 230, non-volatile cache 150, and associated components. In an embodiment, a block contains 100 disk sectors.

In an embodiment of the present invention, the data stored in disk drive 230 may be cached in non-volatile cache 150. In this embodiment, non-volatile cache 150 acts as a special memory subsystem which stores duplicates of frequently used disk drive sectors for quicker access. According to this embodiment, when a disk sector of data is being copied from disk drive 230 into memory 104, a copy of the disk sector may be stored in non-volatile cache 150. According to this embodiment, when the OS determines that it needs data not in memory 104, and the required data is present in non-volatile cache 150, then the data may be read into memory 104 from non-volatile cache 150 (instead of from disk drive 230). In this embodiment, the logical disk drive memory comprises physical disk drive 230 and non-volatile cache 150. If the OS determines that it needs to store data on the disk, for example when the user of a word-processing program instructs the program to "save" the current document, then the data may be stored on the physical disk drive 230 or the non-volatile cache 150. Similarly, when the OS determines it to be necessary to store data that is in memory 104 to free up space in memory 104, the data may be written back to non-volatile cache 150. Because the non-volatile cache 150 is non-volatile, the data saved on it will not be lost if the power is turned off to the computer system.

When non-volatile cache 150 has a faster access time than disk drive 230, the use of non-volatile disk-cache 150 may in the aggregate increase the speed with which processor 102 executes programs. For example, when the average access speed of disk drive 230 is significantly greater than the average access speed for non-volatile disk cache 150, then the use of non-volatile disk cache 150 may lead to a significant increase in overall access speed. Because accesses to the disk drive in a typical computer system may be the cause of up to 80% of the time that a user spends waiting for the system to respond, the use of non-volatile cache 150 will lead to greater user satisfaction. In addition, when non-volatile cache 150 uses less power per access than disk drive 230, then the use of non-

volatile cache 150 may lead to saving in the amount of power used by the system. In addition, non-volatile disk cache 150 may be more reliable than a disk drive, thus minimizing the times when the computer system crashes or becomes inoperable.

In an embodiment, the disk drive's device driver 235 contains cache management instructions 237 for non-volatile cache 150. Such cache management 237 instructions may make decisions regarding what data to cache, what data to replace, and what data to write back to the disk. In addition, cache management instructions 237 may also determine when a cache hit has occurred and what data to pre-fetch into the cache. For example, the cache management instructions 237 may determine whether a desired disk sector of data should be read into memory 104 from non-volatile cache 150 (e.g., if the data is present in non-volatile cache 150) or from disk drive 230. Cache management determinations, such as whether data should be cached in non-volatile cache 150, may be made using known cache algorithms. For example, if the algorithm determines that there is a low likelihood of the data being used again in the near future (e.g., the data is for an MP3 audio file), then the data might not be cached. In addition, when it becomes necessary to write data in non-volatile cache 150 back into disk drive 230, cache management instructions may determine which data to write back using, for example, a known least recently used (LRU) algorithm or a random replacement algorithm.

In an embodiment, the device driver 235 for the disk drive appears to the OS 231 as if it were a normal device driver (e.g., ATAPI.SYS in a WIN98 environment) even though it has cache management instructions 237. In this embodiment, the existence of non-volatile cache 150 is transparent to OS 231. In a further embodiment, the cache management instructions are part of OS 231. In another embodiment, cache management is performed by logic on chipset 110. In a still further embodiment, cache management is performed by a combination of OS 231, the device driver for the disk drive 235, and/or cache management logic on chipset 110.

FIG. 3 is a partial block diagram of a chipset 310 and non-volatile disk cache 350 according to an embodiment of the present invention. Chipset 310 and non-volatile disk

cache 350 may be the same as chipset 110 and non-volatile cache 150 of FIG. 1. In this embodiment, non-volatile disk cache 350 is referred to as a disk cache because it caches data for a disk drive. Non-volatile disk cache 350 may be any type of memory that can be read from and written to and that will retain its contents when all power sources external to the memory are removed or turned off. Non-volatile disk cache 350 may be, for example, a flash memory, a battery backed-up DRAM, a magnetic RAM, a holographic memory, a ferro-electric RAM, etc. In an embodiment, non-volatile disk cache 350 may store 500 megabytes of data. In a further embodiment, non-volatile disk cache 350 is block-oriented, with each block containing one or more logical disk sectors that correspond to logical disk sectors of a disk drive. In this embodiment, each disk sector and block of non-volatile disk cache 350 are the same size as in the disk drive. If disk cache 350 is block-oriented, data (other than cache management information) may be read or stored in terms of a disk sector or block of sectors.

In an embodiment, non-volatile cache 350 is coupled to chipset 310. In a further embodiment, non-volatile disk cache 350 may be directly attached to (i.e., not cabled) or may be part of chipset 310. In an embodiment, shown in FIG. 3, non-volatile cache 150 is stacked on an integrated circuit in chipset 310. In this embodiment, ICH 312 includes disk cache interface logic 315, which controls access to non-volatile disk cache 350.

As discussed above, chipset 310 may be a single integrated circuit or group of integrated circuits that control communication between a processor and associated devices. In an embodiment, chipset 310 comprises multiple integrated circuits, which may be referred to as a first chipset integrated circuit and a second chipset integrated circuit, and the non-volatile disk cache may be coupled to one of the integrated circuits in the chipset. Chipset 310 may include a memory control hub (MCH) 311, which performs what is known as "northbridge functionality," and an input/output controller hub (ICH) 312, which performs what is known as "southbridge functionality." As shown in FIG. 3, the memory control hub 311 and input/output control hub 312 may be separate chips. In an

embodiment, the cache interface logic 315 and a buffer 318 are part of the memory control hub 311 on the chipset. The non-volatile memory may be coupled to the chipset in any manner, and the present invention may be used with any type of chipset. In FIG. 3, the non-volatile memory is shown stacked on top of the input/output control hub 312. The stacked non-volatile memory may cover the entire chip upon which it is stacked or part of the chip.

In an embodiment, a non-volatile cache can be a fully associative cache. In another embodiment, a non-volatile cache can be a set associative cache.

FIG. 4 is a more detailed partial block diagram of a non-volatile disk cache 350 according to an embodiment of the present invention. Non-volatile disk cache 350 may store a plurality of disk cache entries 400 and management information. In an embodiment, each of the disk cache entries stored in non-volatile disk cache 350 is the size of a disk sector on mass storage 220. In a further embodiment, each disk cache entry corresponds to a block of one or more disk sectors (e.g., 100 disk sectors).

Non-volatile disk cache 350 also may contain a cache directory table 410 that has a table entry for every disk cache entry. Cache directory table 410 may be used to determine whether a specific disk sector or block of data is present in non-volatile disk cache 350. In one embodiment, the cache directory table indicates whether data corresponding to a disk drive sector is stored in the disk cache. There may be one table entry in cache directory table 410 for each valid disk cache entry that contains, for example, a logical disk sector of data. Each table entry in cache directory table 410 may contain a sector address for a logical disk sector stored in non-volatile disk cache 350. To determine whether a disk sector is present in non-volatile disk cache 350, the cache directory table 410 may be searched using, for example, a known search algorithm. Alternatively, the table entries in cache directory table 410 may be sorted using a hashing algorithm. In an embodiment, the presence of a disk sector in non-volatile disk cache 350 may be confirmed by comparing the desired sector address against the sector address stored in the cache directory table entry. Because cache directory table 410 is stored in

non-volatile memory, the state of the cache (e.g., what is in the cache) is preserved even if power to the computer system is removed or turned off. In an embodiment that has 2,000,000 disk cache entries in non-volatile disk cache 350, there are 2,000,000 table entries in cache directory table 410. In this embodiment, each table entry may be four bytes long and the cache directly table may use eight megabytes of non-volatile memory. In an embodiment, the size of the cache directory table is reduced (and thus the speed of the average cache access increased) by including a block of multiple disk sectors in each disk cache entry.

Non-volatile disk cache 350 also may contain control registers such as a command register 401, an address register 402, and a destination pointer register 403. Command register 401 may store a command received from another device, such as for example a command from processor 102 to retrieve a disk sector that is stored in non-volatile disk cache 350. Address register 402 may store the sector address for the disk sector being read from non-volatile disk cache 350 or written to non-volatile disk cache 350. Destination pointer register 403 may store the location, for example a device such as a RAM, where the disk sector is being written.

FIG. 5 is block diagram of a non-volatile disk cache entry 500 according to an embodiment of the present invention. Non-volatile disk cache entry 500 may be one of the plurality of disk cache entries 400 shown in FIG. 4 and may contain data for a disk sector or block of disk sectors. Non-volatile disk cache entry 500 may have a valid field 501, a modified field 502, a sector address 505, a data field 510, and a ECC field 520. Valid field 502 may be set to "valid" when cache entry 500 contains valid data and may be set to "invalid" when cache entry 500 does not contain valid data. For example, when the data in cache entry 500 is written back into disk drive 230, valid field 501 may be changed from "valid" to "invalid" to signify that the cache entry no longer stores the logical disk sector. Modified field 502 may be set to "modified" if the data in cache entry 500 is not identical to the data in the corresponding disk sector(s) on disk drive 230. For example, if

a disk sector of data is written back from memory 104 into non-volatile cache 150, and that disk sector contains data that was modified while in memory 104, then the modified field for the disk cache entry corresponding to that data will be set to "modified." Modified field 502 may be referred to as a "dirty bit." Sector address 505 may contain the logical disk sector address (or addresses if the entry has more than one sector) for the data stored in the cache entry and may be referred to as a "sector identifier." In an embodiment, sector address 505 contains the start address for the block, and each block has a known fixed size. Data field 510 stores the data for that cache entry. In an embodiment, data field 510 may contain a disk sector of data (e.g., 512 bytes). ECC field 520 stores error correcting codes for the cache entry. In an embodiment, each block is associated with an error correcting code.

In operation, processor 102 may determine that a certain block of data is not in memory 104 and may signal chipset 110 to determine if the data is in the non-volatile disk cache. Disk cache interface logic 315 may be used to determine, based on cache directory table 410, that the block of data is in the non-volatile disk cache. In order to read the block of data from non-volatile disk cache 350, processor 102 may then send a "read" command to command register 401, the disk sector address for the block to address register 402, and a destination address in memory 104 to destination pointer register 403. Based on this command, disk cache interface logic 315 may write the data found in the corresponding cache entry (e.g., the data in data field 510 of cache entry 500) into buffer 318. The data in the buffer may then be transferred to the location in memory 104 specified by the destination pointer register 403. In another embodiment, because of the speed of the memory, a buffer 318 may not be used. In an embodiment, disk cache interface logic 315 may check error correcting codes associated with data stored in the non-volatile memory (e.g., ECC codes 520) to correct errors that may occur when reading the data from the non-volatile disk cache memory into buffer 318. In an embodiment, the

non-volatile memory is block-oriented and the chipset checks error correcting codes for each block of data read from the non-volatile memory.

In an embodiment, reads from the non-volatile disk cache memory are destructive reads, in that the act of reading causes the data stored in that entry to be lost or changed unpredictably. This may occur, for example, if the non-volatile disk cache is implemented in core logic. In this embodiment, disk cache interface logic 315 may support write back of destructive reads. That is, disk cache interface logic 315 may cause the data that is read to be written back, thereby restoring the data in the entry to its prior state.

Alternatively, if the data desired by processor 102 is not present in the non-volatile disk cache, a request for the appropriate block of data may be sent to disk drive 230. Disk drive 130 may write the block of data to memory 104. In addition, it may write the same block of data to the non-volatile disk cache by writing the block to buffer 318 and by writing the appropriate command to command register 401. Cache interface logic 315 may then create a new disk cache entry 500 and may copy the block into the new disk cache entry. In addition, cache interface logic 315 may create a new entry in cache directory table 410 corresponding to the new disk cache entry. The valid field 501 for the new disk cache entry would be set to "valid" and the modified field would be set to "unmodified." In an embodiment, the blocks in the non-volatile disk cache have the same size and structure as blocks in the disk drive, so that a block can be copied into the disk cache without modification or with only minor changes. Once the block of data is stored in the non-volatile disk cache, processor 102 may read the block of data from the non-volatile disk cache instead of from the disk drive.

FIG. 6 is a flow chart of a method of obtaining data associated with a sector in a disk drive according to an embodiment of the present invention. For purposes of clarity, the illustrated method is described in connection with the exemplary embodiments described above. The processor 102 may need an item of data and may determine that the data is stored at, or associated with, a location in a disk drive (601). The most current

version of data associated with a location in the disk drive may be stored in the RAM or the disk cache. If the data is not stored in the RAM, the processor may cause a check of the cache directory table to determine whether data associated with the disk drive location is stored in the non-volatile disk cache (602). If the data is stored in the non-volatile disk cache (603), then the data may be read from the non-volatile disk cache (604). Error correcting codes may then be checked for the data as it is read from the non-volatile disk cache, and errors may be corrected (605). The data may then be written to RAM (606). If the data is not stored in the non-volatile disk cache (603), then the requested data may be read from the disk drive (607). The data may then be written to RAM and to the non-volatile disk cache, thus creating a new entry in the non-volatile disk cache (608). Next, a new entry in the cache directory table may be created corresponding to the new entry in the non-volatile disk cache (609).

FIG. 7 is a partial block diagram of a computer system that has a non-volatile cache expansion card according to another embodiment of the present invention. In this embodiment, a non-volatile disk cache is contained on a PCI bus expansion card. FIG. 7 shows a computer system 700 that may be similar to the computer system 100 shown in FIG. 1. FIG. 7 shows a motherboard 706 that contains a CPU 702, RAM 704, chipset 710, and disk drive connector cable 770 that may be the same as CPU 102, memory 104, chipset 110 and connector cable 170 of FIG. 1. Computer system 700 also contains a disk drive 730 which may be the same as mass storage device 130 of FIG. 1. Unlike in FIG. 1, however, in the embodiment of FIG. 7 a non-volatile cache is not attached to chipset 710.

In this embodiment of the present invention, computer system 700 contains a non-volatile cache expansion board 750 that is attached to a PCI bus 716. A network interface card 726 is also attached to PCI bus 716. PCI bus 716 and network interface card 726 may be the same as expansion bus 116 and network interface card 126 of FIG. 1. In other embodiments, non-volatile cache expansion board 750 may be attached to another expansion bus in computer 700. Non-volatile cache expansion board 750 may be plugged

in or removed from a bus of computer 700. Non-volatile cache expansion board 750 contains non-volatile memory that may be similar to the non-volatile memory in non-volatile disk cache 350. Expansion board 750 may also contain disk cache interface logic. Computer 700 may cache data for disk drive 730 in a similar fashion as described above with reference to computer system 100. When the operating system in computer 700 needs to obtain data that may be stored on disk drive 730, a determination may be made that the data is being cached in non-volatile cache expansion board 750. If so, then the data may be written to RAM 704 from non-volatile expansion board 750 instead of disk drive 730. Thus, this embodiment is similar to the embodiment of FIG. 1 in the way it caches data for the disk drive, although in this embodiment the cache is part of an expansion card rather than attached to the chipset as in FIG. 1.

FIG. 8 is a partial block diagram of a disk drive that has a non-volatile disk cache according to an embodiment of the present invention. FIG. 8 shows a disk drive 830 which may be similar to the disk drive 230 shown in FIG. 2. Like disk drive 230, disk drive 830 may have a disk controller 810 and a disk spindle 820. Disk drive 830 also has a non-volatile cache 850. Mass storage 220 may store an operating system 831 and a disk drive device driver 835. However, disk spindle 820 does not store cache management instructions. Disk controller 810 has a MPU 811, buffer and program memory 815, interface logic 813, and ECC logic 818 which may be similar to MPU 211, buffer and program memory 215, disk spindle interface logic 213, and ECC logic 218 of FIG. 2. In addition, disk controller 810 has cache interface logic 815 and is coupled to non-volatile cache 850. Non-volatile disk cache 850 may be similar to non-volatile cache 350 of FIG. 3, and cache interface logic 815 may be similar to disk cache interface logic 315 shown in FIG. 3.

Disk drive 830 may be used in a computer system such as that shown in FIGS. 1 and 7, except that non-volatile cache 850 may be the only non-volatile disk cache in the computer system. The embodiment shown in FIG. 8 operates similarly to the

embodiments of FIGS. 1 and 7 in the that it caches data for the disk drive in a non-volatile memory. In this embodiment, however, the disk cache may be part of the disk drive. In a further embodiment, the disk cache management function may be performed by program 819. In this embodiment the disk cache may be managed by the disk controller 810 and may be transparent to the rest of the computer system.

FIG. 9 shows a method of storing data in a non-volatile memory according to an embodiment of the present invention. A command to store data in non-volatile memory is received (901). For example, a user who is editing a document in a word processing program may send the program a command to "save" the document. As another example, the program may on its own determine, for example based on a timer, that a backup copy of the document being edited should be saved. As another example, a part of the operating system may generate a command to store data into a disk drive or other non-volatile memory, for example to free room in the RAM. A determination is made whether to store the data in a non-volatile cache (902). Such a determination may be made using a known caching algorithm. Data may not be cached, for example, if it is not likely to be used in the near future. In addition, a determination may be made not to store data in a non-volatile cache if a non-volatile cache expansion board is not coupled to an expansion bus. When a determination is made to store data in the non-volatile cache, the data is written to a non-volatile cache (903). In one embodiment, writing data to the non-volatile cache includes updating a cache directory table in a non-volatile memory on the non-volatile cache (904). When a determination is made not to store data in the non-volatile cache, the data is written to a disk drive (905).

Embodiments of the present invention relate to a non-volatile disk cache to cache data for a disk storage device. Several embodiments of the present invention are specifically illustrated and/or described herein. However, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and

intended scope of the invention. For example, although several types of non-volatile memory are discussed for use as the cache, any type of non-volatile memory may be used. As another example, although the present invention discloses a data structure for a non-volatile disk cache and for a disk cache entry, the cache and cache entries may be implemented using any configuration of data structure. Moreover, the disk cache may be implemented as part of a chipset, as an extension card, as part of the disk drive, or in another part of the computer system, such as within a disk drive connector cable. In addition, while the embodiments described above discuss using the non-volatile cache to cache data for a disk drive, the non-volatile cache may also be used to cache data for other mass storage devices. Thus, while the embodiments shown such as in FIGS. 3, 4 and 5 for convenience discuss the cache with reference to cache as a disk drive (i.e., a disk cache), the apparatus and methods disclosed may also be used to cache data for other types of mass storage devices.

I CLAIM:

1. A non-volatile cache, comprising a non-volatile memory to cache data that is stored on a mass storage device.

2. The non-volatile cache of claim 1, wherein the mass storage device is a disk drive.

3. The non-volatile cache of claim 1, wherein the non-volatile memory is block-oriented.

4. The non-volatile cache of claim 1, wherein the non-volatile memory is coupled to a chipset.

5. A disk cache, comprising:

a block-oriented memory device; and

disk cache interface logic coupled to the block-oriented memory device.

6. The disk cache of claim 5, wherein the memory device is a non-volatile memory device.

7. The disk cache of claim 6, wherein the memory device is to store a cache directory table, the cache directory table to indicate whether data corresponding to a disk drive sector is stored in the disk cache.

8. The disk cache of claim 7, wherein the disk cache interface logic is to check error correction codes for data read from the non-volatile memory device.

9. An apparatus to cache data, the apparatus comprising a chipset coupled to a non-volatile cache memory.

10. The apparatus of claim 9, wherein the chipset contains logic to control communication between a processor and an associated device.

11. The apparatus of claim 9, wherein the non-volatile cache memory is a disk cache memory.

14. The apparatus of claim 11, wherein the non-volatile cache memory is directly connected to the chipset.

15. The apparatus of claim 12, wherein the non-volatile cache memory is block-oriented.

16. The apparatus of claim 11, wherein the non-volatile cache memory is part of the chipset.

17. The apparatus of claim 14, wherein the non-volatile cache memory is stacked on the chipset.

18. The apparatus of claim 11, wherein the chipset includes cache interface logic to control access to the non-volatile cache memory.

19. The apparatus of claim 16, wherein the cache interface logic is part of a memory control hub on the chipset.

1. 20. The apparatus of claim 16, wherein the chipset further includes logic to check error correcting codes associated with data stored in the non-volatile cache memory.

21. The apparatus of claim 18, wherein the non-volatile memory is block-oriented, and wherein the apparatus contains logic to check error correcting codes for each block of data read from the non-volatile memory.

22. A computer, comprising:

a processor;

a chipset coupled to the processor;

a disk storage device coupled to the chipset; and

a non-volatile disk cache memory coupled to the chipset.

23. The computer of claim 20, further comprising device driver instructions to store data on the disk storage device and read data from the disk storage device, said device driver instructions including cache management instructions to manage caching of data in the non-volatile disk cache memory.

24. The computer of claim 21, wherein the disk memory device driver instructions include instructions to determine whether the data is present in the non-volatile disk cache memory by checking a cache directory table stored in the non-volatile disk cache memory.

25. The computer of claim 22, wherein the non-volatile disk cache is block-oriented.

26. The computer of claim 23, wherein:

the chipset includes disk cache interface logic to control access to the non-volatile disk cache; and

the disk cache interface logic is to check error correction codes associated with blocks of data read from the non-volatile disk cache.

27. A method of obtaining data associated with a location in a disk memory, the method comprising:

determining whether data associated with a location in a disk memory is stored in a non-volatile cache memory; and

reading data associated with the location from the non-volatile cache memory when said data is stored in the non-volatile cache memory.

28. The method of claim 25, further comprising:

reading data associated with the location from the disk memory when said data is not stored in the non-volatile cache memory.

29. The method of claim 26, wherein said determining comprises checking a cache directory table stored in the non-volatile memory.

30. The method of claim 27, wherein the non-volatile cache memory is coupled to a chipset, and wherein said reading data from the non-volatile cache memory includes sending a command to cache interface logic located within the chipset, the command to direct reading of the data from the non-volatile cache memory into a RAM.

31. The method of claim 28, wherein the non-volatile cache memory is block-oriented, and wherein reading data from the non-volatile cache memory further comprises checking error correcting codes associated with each block read from the non-volatile cache memory.

32. The method of claim 29, wherein said reading data associated with the location from the disk memory further comprises storing said data to a RAM and storing said data to the non-volatile cache memory.

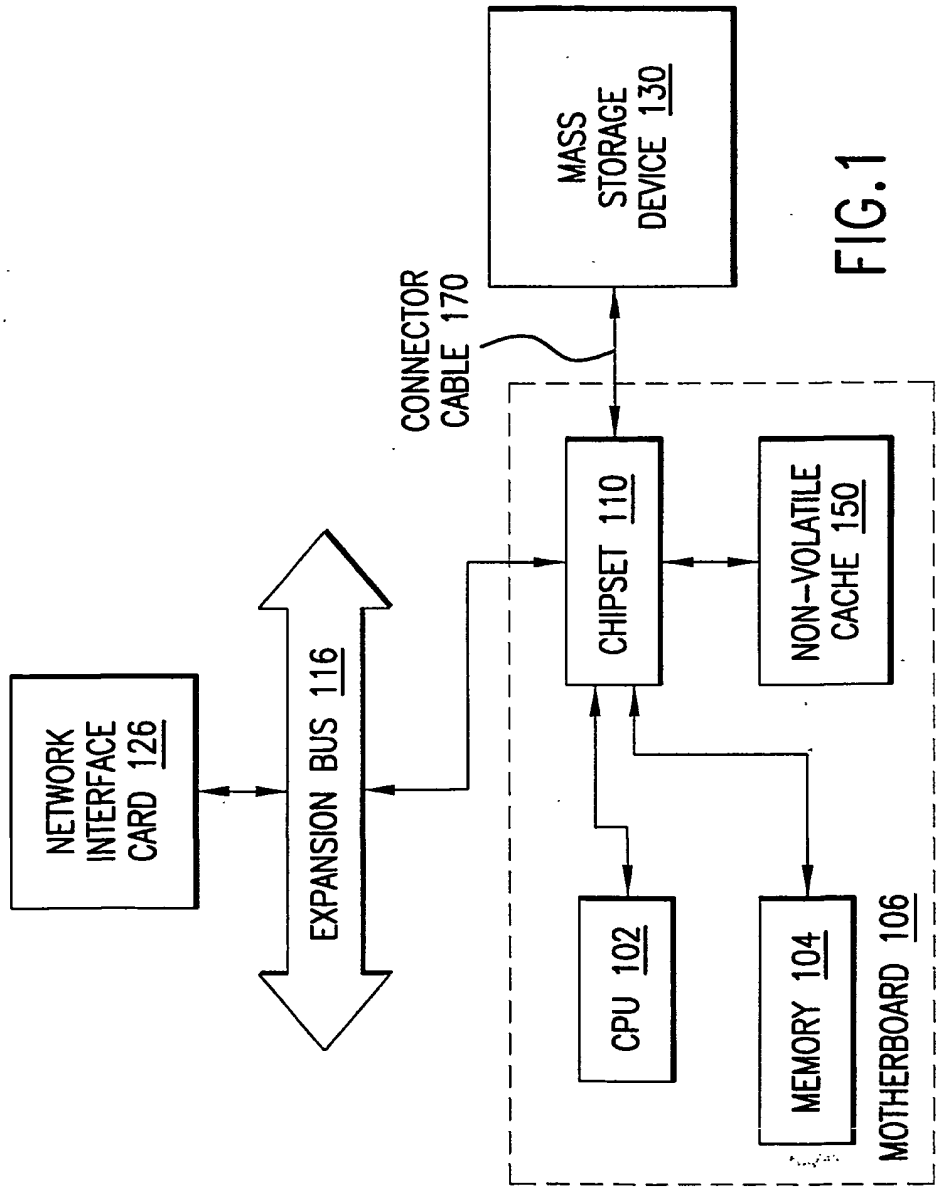

MASS STORAGE DEVICE 130
CONNECTOR CABLE 170
CHIPSET 110
NON-VOLATILE CACHE 150
NETWORK INTERFACE CARD 126
EXPANSION BUS 116
CPU 102
MEMORY 104
MOTHERBOARD 106

FIG. 1

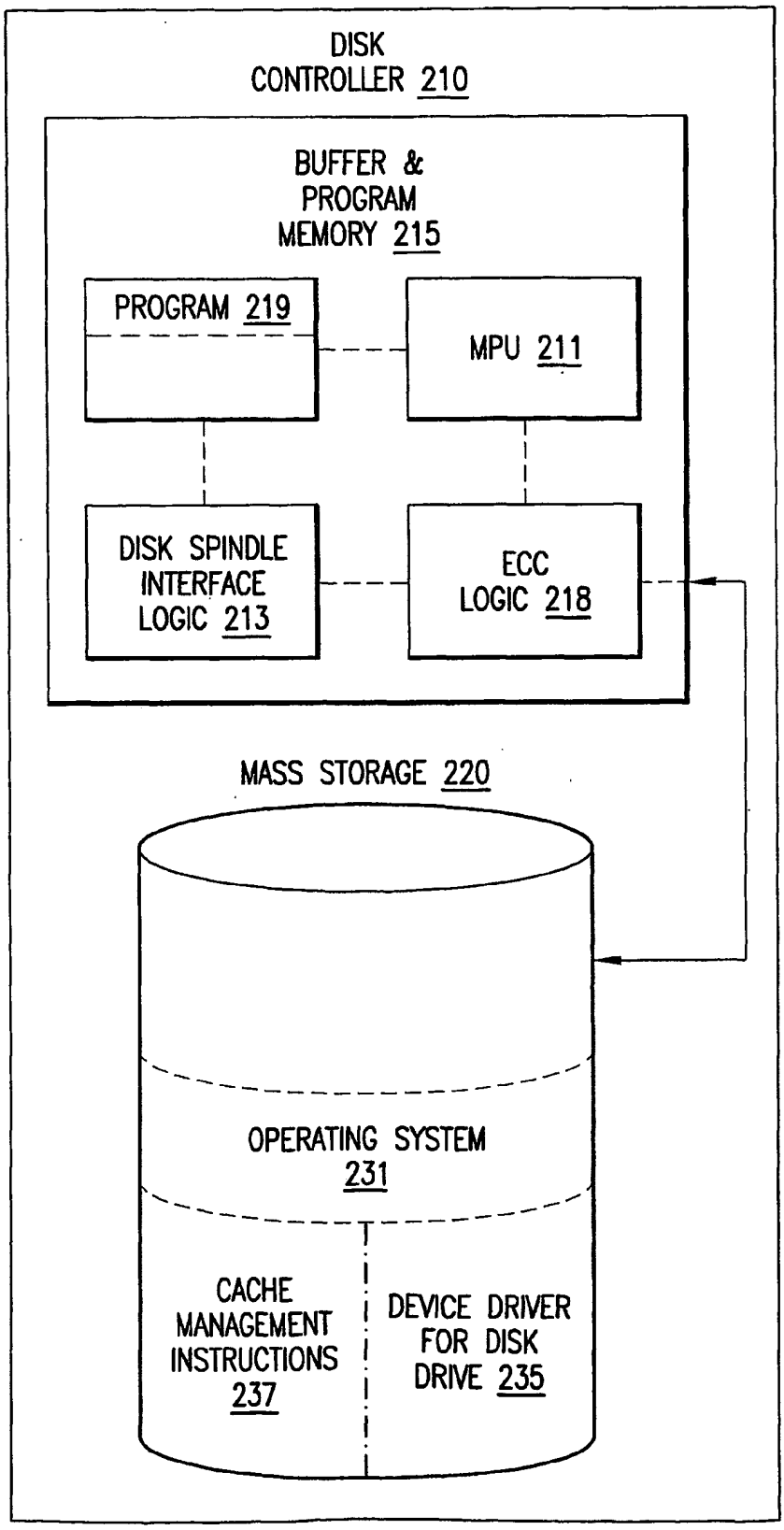

DISK
CONTROLLER 210
BUFFER &
PROGRAM
MEMORY 215
PROGRAM 219
MPU 211
DISK SPINDLE
INTERFACE
LOGIC 213
ECC
LOGIC 218
MASS STORAGE 220
OPERATING SYSTEM
231
CACHE
MANAGEMENT
INSTRUCTIONS
237
DEVICE DRIVER
FOR DISK
DRIVE 235
DISK DRIVE 230

FIG.2

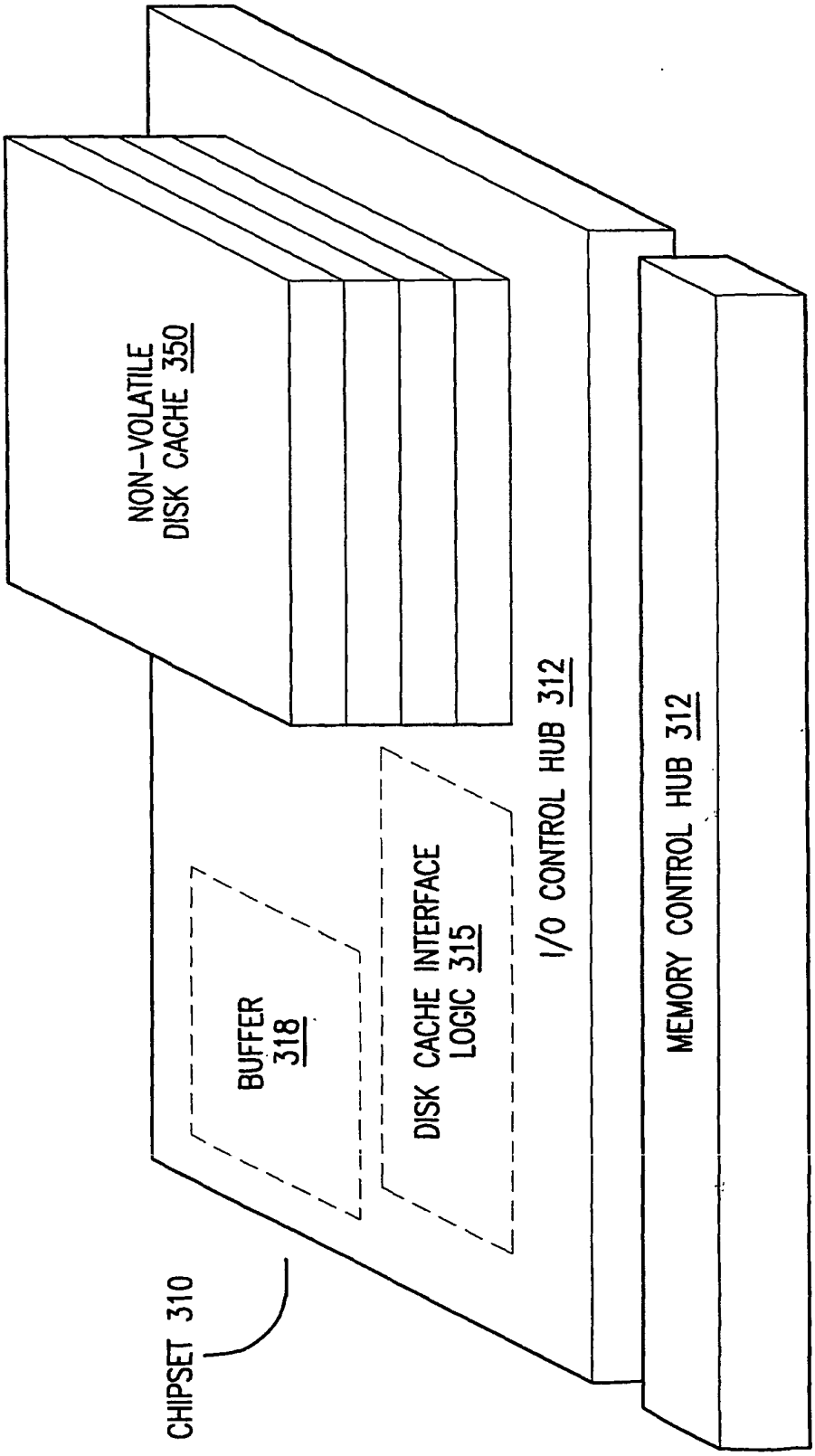

NON-VOLATILE
DISK CACHE 350
I/O CONTROL HUB 312
MEMORY CONTROL HUB 312
BUFFER
318
DISK CACHE INTERFACE
LOGIC 315
CHIPSET 310

FIG.3

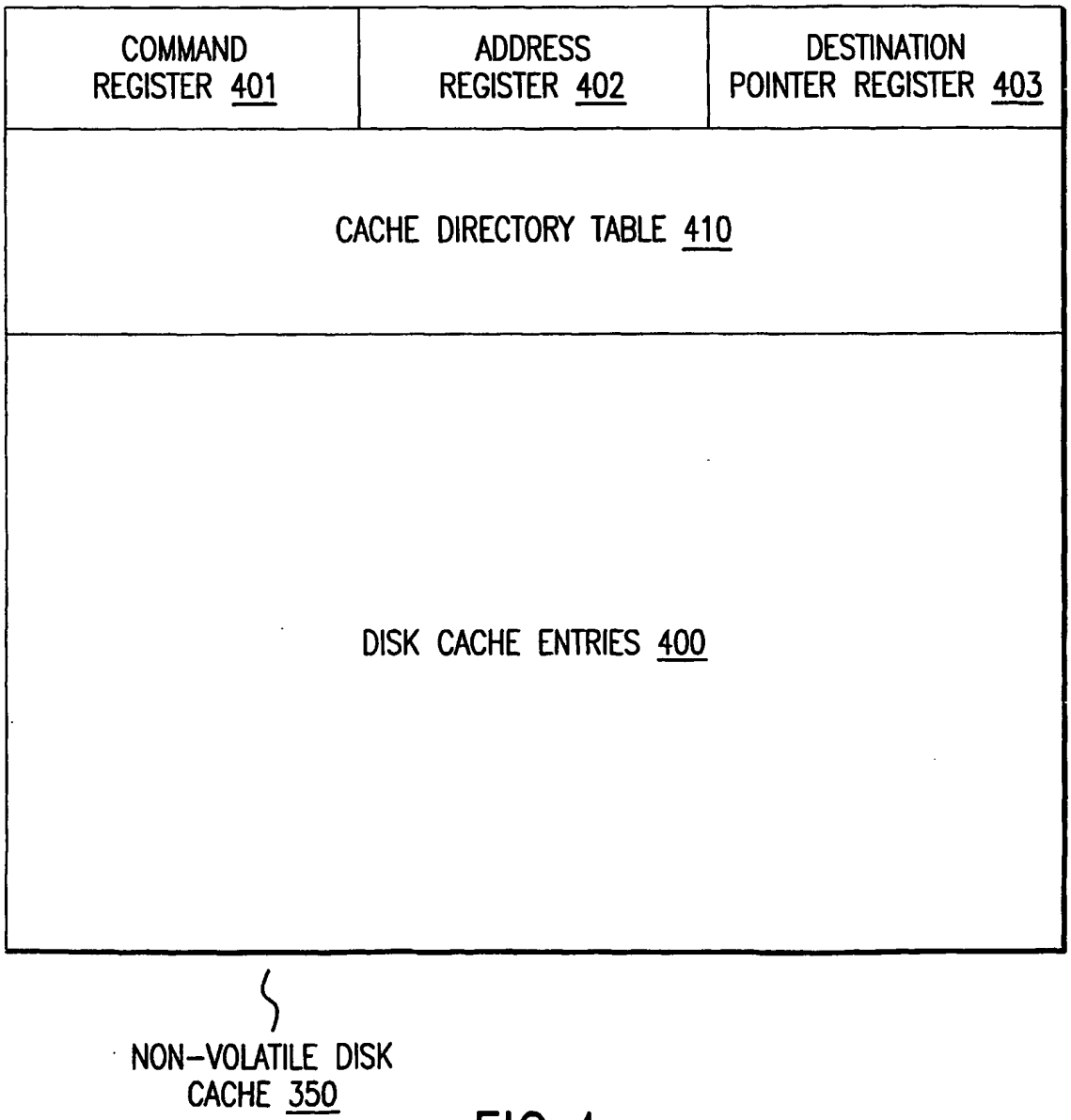

COMMAND REGISTER 401
ADDRESS REGISTER 402
DESTINATION POINTER REGISTER 403
CACHE DIRECTORY TABLE 410
DISK CACHE ENTRIES 400
NON-VOLATILE DISK CACHE 350

FIG.4

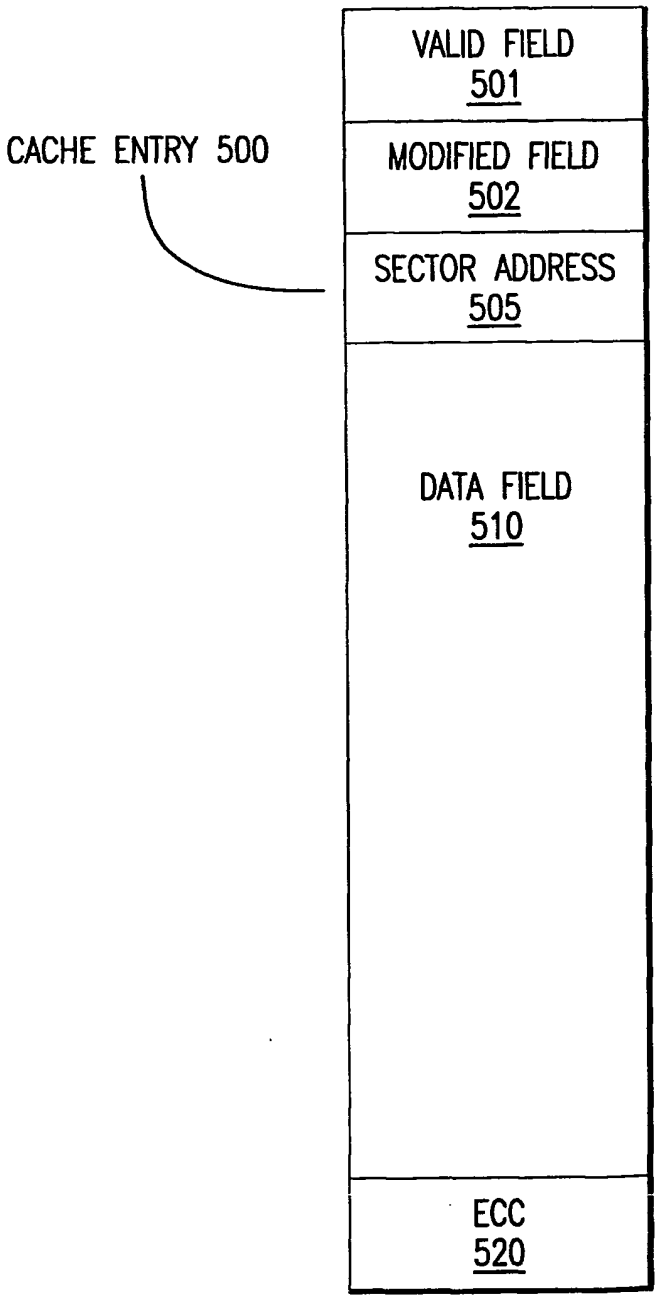

CACHE ENTRY 500
VALID FIELD
501
MODIFIED FIELD
502
SECTOR ADDRESS
505
DATA FIELD
510
ECC
520

FIG.5

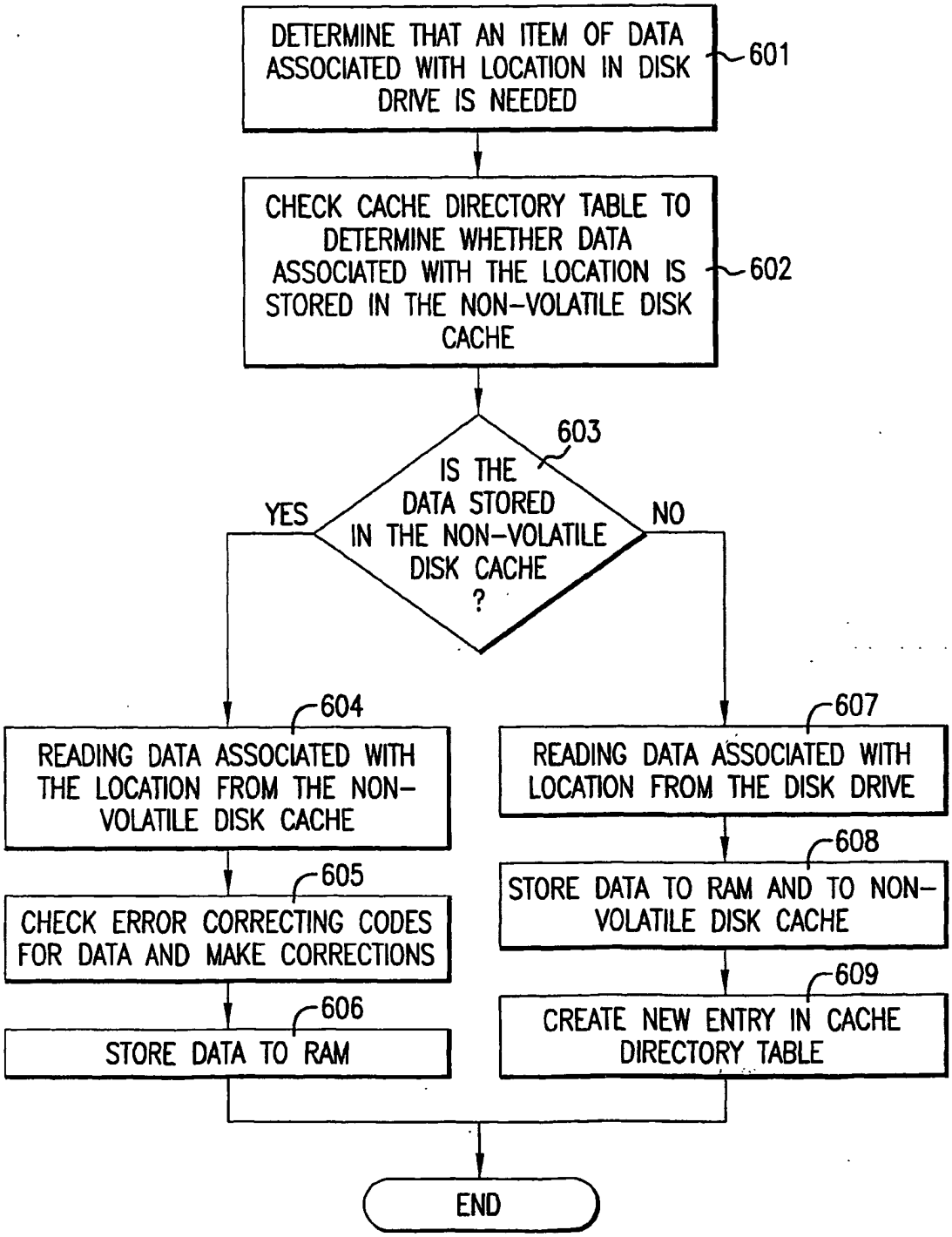

DETERMINE THAT AN ITEM OF DATA ASSOCIATED WITH LOCATION IN DISK DRIVE IS NEEDED
601
CHECK CACHE DIRECTORY TABLE TO DETERMINE WHETHER DATA ASSOCIATED WITH THE LOCATION IS STORED IN THE NON-VOLATILE DISK CACHE
602
603
IS THE DATA STORED IN THE NON-VOLATILE DISK CACHE ?
YES
NO
604
READING DATA ASSOCIATED WITH THE LOCATION FROM THE NON-VOLATILE DISK CACHE
605
CHECK ERROR CORRECTING CODES FOR DATA AND MAKE CORRECTIONS
606
STORE DATA TO RAM
607
READING DATA ASSOCIATED WITH LOCATION FROM THE DISK DRIVE
608
STORE DATA TO RAM AND TO NON-VOLATILE DISK CACHE
609
CREATE NEW ENTRY IN CACHE DIRECTORY TABLE
END

FIG.6

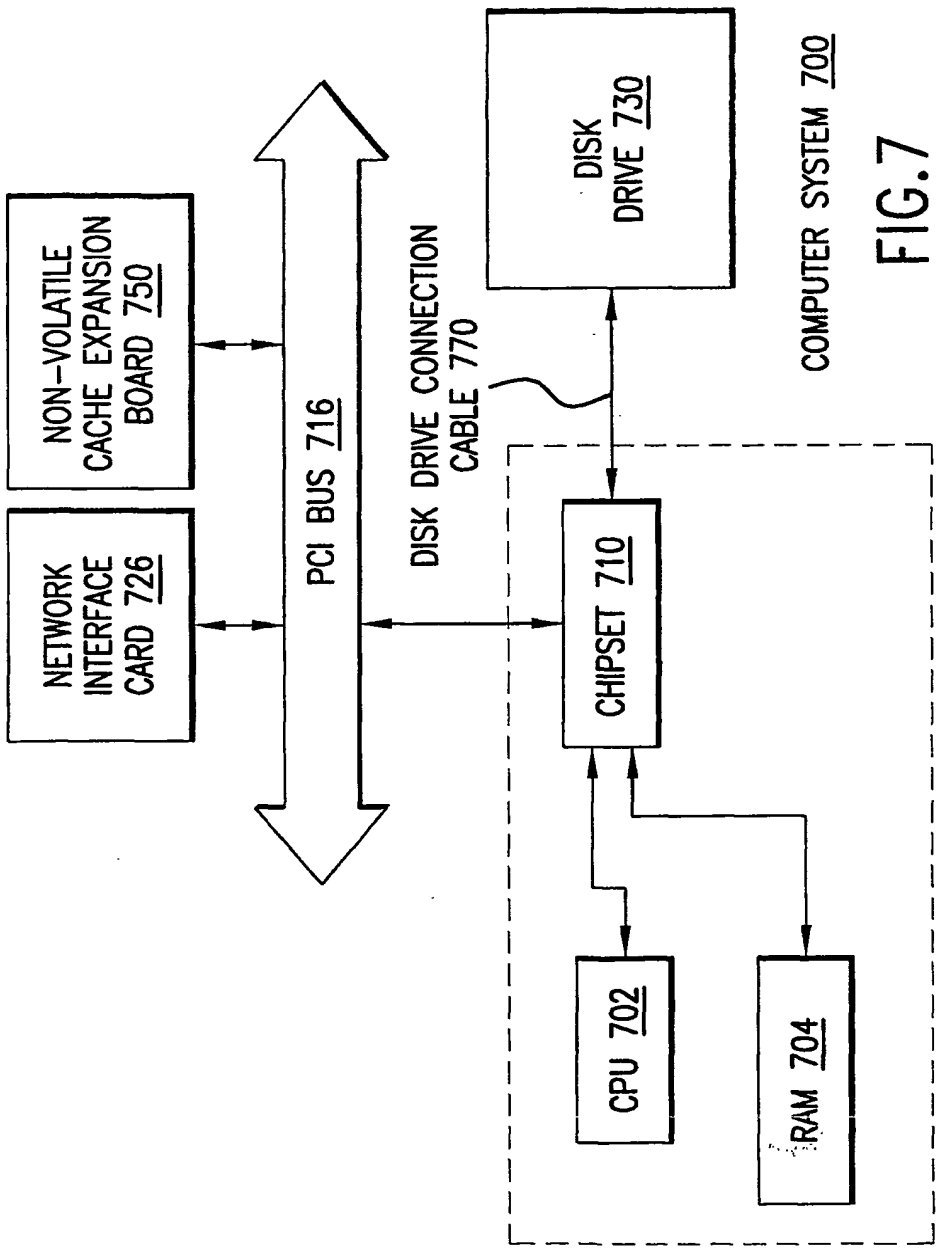

NON-VOLATILE CACHE EXPANSION BOARD 750
NETWORK INTERFACE CARD 726
PCI BUS 716
DISK DRIVE CONNECTION CABLE 770
DISK DRIVE 730
COMPUTER SYSTEM 700
CHIPSET 710
CPU 702
RAM 704

FIG.7

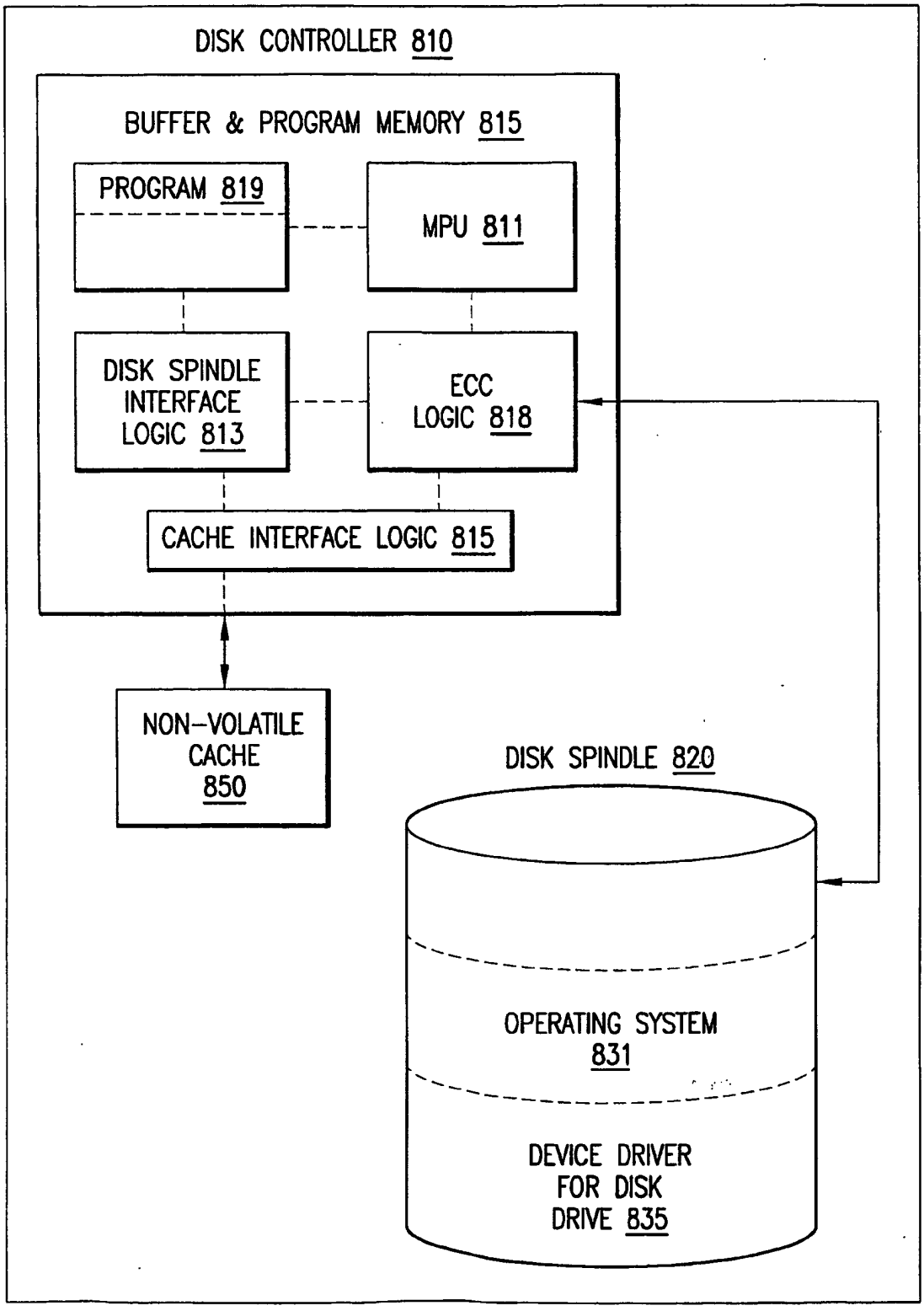

DISK CONTROLLER 810
BUFFER & PROGRAM MEMORY 815
PROGRAM 819
MPU 811
DISK SPINDLE INTERFACE LOGIC 813
ECC LOGIC 818
CACHE INTERFACE LOGIC 815
NON-VOLATILE CACHE 850
DISK SPINDLE 820
OPERATING SYSTEM 831
DEVICE DRIVER FOR DISK DRIVE 835
DISK DRIVE 830

FIG.8

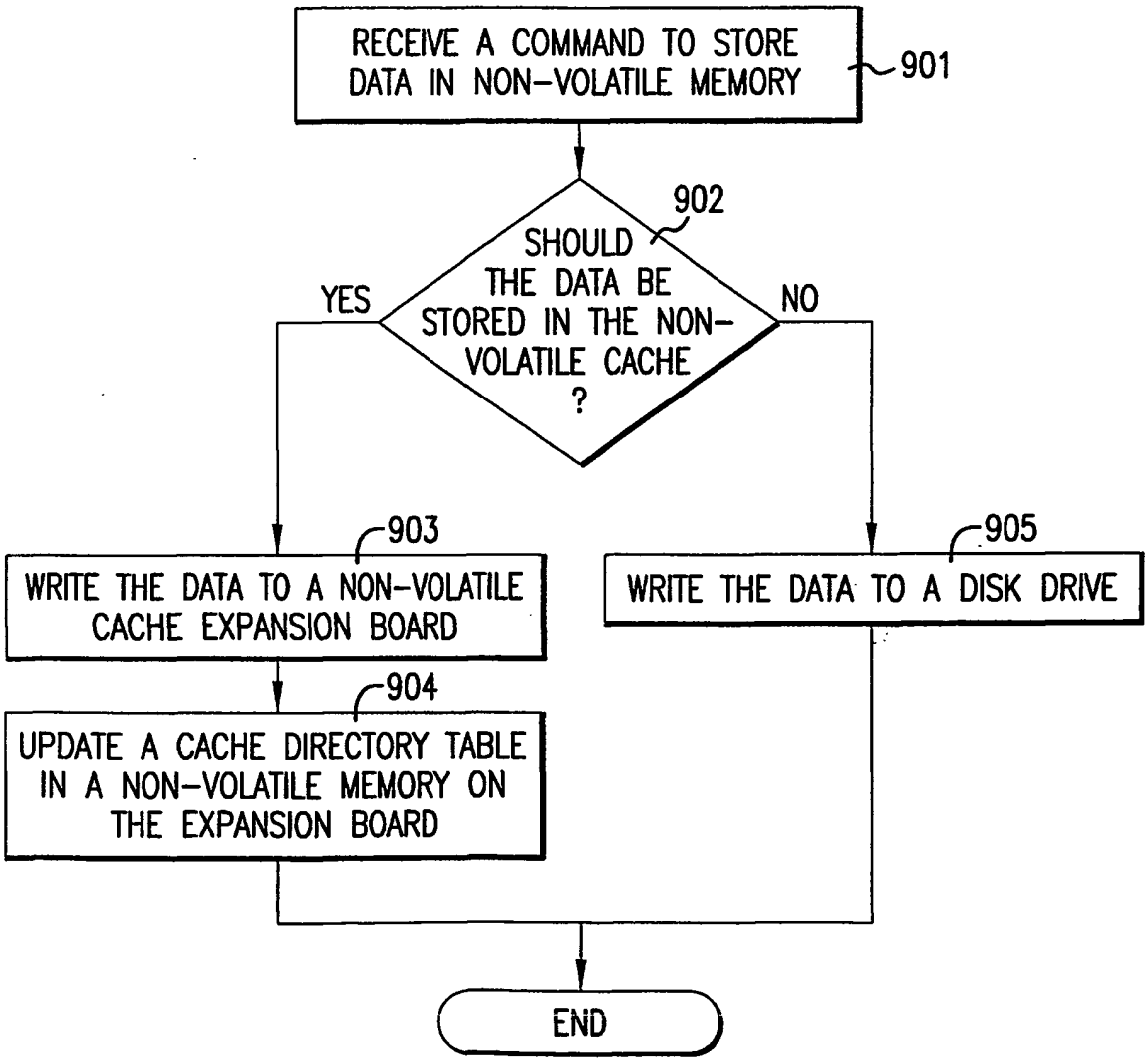

RECEIVE A COMMAND TO STORE DATA IN NON-VOLATILE MEMORY
901
902
SHOULD THE DATA BE STORED IN THE NON-VOLATILE CACHE ?
YES
NO
903
WRITE THE DATA TO A NON-VOLATILE CACHE EXPANSION BOARD
904
UPDATE A CACHE DIRECTORY TABLE IN A NON-VOLATILE MEMORY ON THE EXPANSION BOARD
905
WRITE THE DATA TO A DISK DRIVE
END

FIG.9